# Tentamen i EDAF75

## 18 mars 2025

## Skrivtid: 14:00-19:00

- Skriv tydligt: om texten inte går att läsa kan du inte få några poäng.
- Skriv bara på ena sidan av pappret: tentorna kommer att scannas in, och endast framsidorna rättas.
- Skriv ingen personlig kod på tentan: den 'personliga kod' som används på en del tentor gör tentorna mindre anonyma, så vi vill inte ha den!
- Sätt identitet och sidnummer på varje inlämnat blad: om din anonymkod skulle vara `EDAF75-0123-ABC` så skall du bara skriva `123-ABC` längst upp i högerkanten på varje inlämnat blad (hoppa över kurskod och inledande nollor) – skriv dessutom sidnumret och antal inlämnade blad under identiteten, som nedan (exemplet visar sidan 3 av 5 på tentan med anonymkoden `EDAF75-0123-ABC`):

  `123-ABC`
  3/5

- Kontrollera noga att du får med alla blad, och lägg dem i rätt ordning i omslaget. Om du markerar sidorna enligt ovan, *och lägger dem i ordning*, så är det enklare att kontrollera att alla sidor finns med.
- *Preliminär* maxpoäng anges vid varje uppgift.
- *Preliminära* betygsgränser:

  3: Minst ca 60% på *var och en* av uppgift 1-3 (alla tre måste vara godkända).

  4: Minst ca 70% på *var och en* av uppgift 1-3, och ca 70% totalt på uppgift 4-6.

  5: Minst ca 85% på *var och en* av uppgift 1-3, och ca 85% totalt på uppgift 4-6.

## Uppgift 1

I denna uppgift förekommer ett antal begrepp som kan vara svåra att översätta till engelska under tentan, om du vill så får du gärna använda svenska namn i din modell (å, ä och ö går bra!).

Ett galleri vill ha en relationsdatabas för att administrera sin verksamhet. På galleriet finns ett antal tavlor, varje tavla har en titel, och är målad av en konstnär ett givet år. Tavlorna kan klassificeras i olika kategorier, och varje tavla kan tillhöra flera kategorier (olja, abstrakt, 1920-tal) – varje kategori har ett unikt namn, och en kort beskrivning.

En konstnär har ett unikt namn, ett födelseår, ett (eventuellt) dödsår, och ett telefonnummer. En konstnär kan ha flera av sina tavlor på galleriet, och vi antar att konstnären kan använda samma titel på flera av sina tavlor, dock används en titel högst en gång per år för en given konstnär.

Galleriet har även ett antal kunder, alla kunder har namn och email-adresser, vi antar att bara email-adresserna är unika.

Galleriet arrangerar ibland auktioner för enstaka tavlor, en auktion hålls vid en given tidpunkt, den gäller *en* specifik tavla, och har ett minimipris – om ingen bjuder över minimipriset kommer auktionen att ställas in (så att man eventuellt kan ha en ny auktion med samma tavla vid ett senare tillfälle). Det är möjligt att en tavla säljs vid flera auktioner, oavsett om den blivit såld eller inte vid tidigare auktioner (galleriet säljer den på uppdrag av dess nuvarande ägare).

Vid auktionerna lägger kunderna bud – varje kund kan lägga flera bud vid en given auktion, och den som lagt det högsta budet när auktionen avslutas får köpa tavlan, om minimipriset uppnåtts. Vi vill lagra alla bud i databasen, så att vi senare kan göra sökningar och analyser.

(a) Gör en E/R-modell som beskriver databasen, redovisa med ett UML-diagram, markera nycklar, och skriv ut multipliciteter på samtliga associationer. *Skriv inte ut främmande nycklar som redan finns markerade i diagrammet.* Du behöver inte markera dina entity-sets med "`«weak»`".

Skriv SQL-satser för att definiera samtliga tabeller för din E/R-modell – var noga med att deklarera primärnycklar och främmande nycklar – du behöver inte skriva med stora bokstäver, och du behöver inte göra `DROP TABLE` innan du skapar en tabell.

Du får gärna använda samma svenska namn på tabeller och attribut som du använt i din E/R-modell (se ovan). *11 p*

(b) Utgående från dina tabeller i uppgift (a), skriv en SQL-sats som skriver ut namnet på alla de kunder som lagt bud på tavlor av konstnären Oddput Clementin. *2 p*

## Uppgift 2

En fågelskådarförening har en databas med uppgifter om klubbens medlemmar och de fågelarter som medlemmarna har observerat på olika platser. Detta beskrivs av följande databasschema (understrukna attribut är delar av primärnycklar, ***kursiverade fetstilta*** attribut är främmande nycklar):

```
members(member_id, member_name, member_phone)
species(species_id, latin_name, swedish_name)
sites(site_id, site_name, latitude, longitude)
observations(observation_id, member_id, species_id, site_id, observation_date)
```

Alla primärnycklar ovan[1] är 'invented keys'. Attributen `latin_name` och `swedish_name` är de latinska respektive svenska namnen på arten (exempelvis 'Fringilla coelebs' och 'bofink') – vi kan förutsätta att dessa namn är unika.

I tabellen `observations` är attributet `observation_date` det *datum* som en observation gjorts, vi bryr oss i uppgiften inte om hur dags på dagen observationen gjordes.
Rita ett diagram i uppgift (a), och skriv SQL-satser för uppgift (b) - (e) – databasen har precis de primärnycklar och främmande nycklar som är angivna ovan:

(a) Rita ett ER-diagram i UML-format för att beskriva databasen, rita diagrammet enligt anvisningarna i problem 1a. *2 p*

(b) Skriv ut platsnamn och datum för alla observationer av 'bofink', sorterade i första hand efter platsnamn, i andra hand efter datum. *2 p*

(c) Skriv ut de latinska namnen på de arter i databasen som inte observerats någon gång under 2024 – du kan använda funktionen `year` för att få året för ett datum. *3 p*

(d) Skriv ut hur många observationer av de olika arterna som medlemmen 'Ulrika Vogel' har gjort – du kan förutsätta att det bara finns bara en Ulrika Vogel i föreningen. Listan skall sorteras efter antal observationer, och bara innehålla arter som hon har observerat minst 10 gånger:

- talgoxe 24
- blåmes 22
- pilfink 18

*3 p*

(e) För varje art som har observerats, skriv ut följande information om den/de senaste observationen/erna (alla observationer där datum är lika med datum för den senaste observationen av arten skall skrivas ut): svenskt artnamn, platsnamn, observatörsnamn och datum (för den senaste observationen). Ordna utskriften efter de svenska artnamnen, arter som inte har observerats skall inte tas med. *3 p*

[1] De slutar alla på `_id`.

## Uppgift 3

För att ingen skall tappa onödiga poäng på ett eventuellt misstag i (a), så har vi helt separerat uppgift (a) från uppgift (b)..(e).

(a) I ett system för hantering av studieresultat har vi följande attribut:

- `stil_id`: stil-id (unikt för varje student)
- `ssn`: personnummer (unikt för varje student)
- `student_name`: namn på student (behöver inte vara unikt)
- `course_code`: kod för en given kurs (unikt för varje kurs)
- `course_name`: namn för en given kurs (behöver inte vara unikt)
- `grade`: godkänt betyg för en kurs – betyget kan vara 3, 4 eller 5 (en student kan i uppgiften bara bli godkänd en gång på en kurs)
- `exam_date`: datum för ett godkänt betyg (datum då tentamen gavs)

Bestäm de funktionella beroenden som gäller för attributen ovan, och *använd beteckningarna ovan*, dvs introducera *inte* nya namn som `A`, `B`, etc. För full poäng vill vi att dina beroenden skall vara så enkla som möjligt (men vi gör bara små avdrag om du lägger till 'onödiga' men korrekta attribut i högerleden). *4 p*

(b) I relationen $R(A, B, C, D, E, F)$ gäller följande funktionella beroenden:

$\text{FD}_1$: $CE \longrightarrow FD$

$\text{FD}_2$: $AC \longrightarrow B$

Förklara kortfattat begreppen *nyckel* och *supernyckel*, och bestäm nycklarna för relationen $R$ (motivera ditt svar). *2 p*

(c) Visa att relationen inte är i BCNF. *1 p*

(d) Bryt ner relationen $R$ i (b) i mindre relationer som är i BCNF, och som kan rekonstrueras till den ursprungliga relationen utan att vi förlorar eller lägger till rader. Motivera nedbrytningen genom att *i varje steg* skriva dina relationer på nedanstående form, och *förklara de uppdelningar du gör*:

| $R(A, B, C, D, E, ...)$ |
|---|
| Beroenden: $A \longrightarrow B$, $BC \longrightarrow D$, ... |
| Nycklar: $AC$, $BD$, ... |

Det vi är intresserade av är alltså framförallt stegen som leder fram till de slutliga relationerna, *att svara genom att bara ange relationer utan att motivera nedbrytningarna ger inga poäng.* *5 p*

(e) Visa med en SQL-sats hur vi kan återskapa den ursprungliga relationen i (b) med hjälp av relationerna som du bröt upp den i ovan. Denna uppgift ger bara poäng om lösningen i (d) motiverar att vi får tillbaka exakt samma rader. *1 p*

## Uppgift 4

Skriv en trigger som ser till att vi i databasen i uppgift 2 inte kan lägga in mer än tre observationer av en given art på en given plats en given dag, oavsett vem som gör observationerna. *4 p*

## Uppgift 5

Skriv ett Python- eller Javaprogram som med samma databas som i uppgift 4, alltså en lätt modifierad variant av den i uppgift 2, löser nedanstående uppgift – vi antar att triggern i uppgift 4 är inlagd, oavsett om du löst uppgift 4 eller inte.

> Vi vill läsa in data för en observation, och lägga in den i databasen. För att göra uppgiften lite enklare antar vi att både `member_id` och `site_id` är heltal, och att vi läser in nycklarna direkt i programmet. För arten läser vi in det latinska namnet som en sträng.
>
> Lägg därefter in motsvarande observation i databasen – nyckeln (`observation_id`) och datumet (`observation_date`) får båda automatiskt startvärden när vi lägger in en rad i tabellen `observations`, datumet sätts till dagens datum.
>
> Skriv sedan ut namn och telelfonnummer för samtliga medlemmar som observerat samma art på samma ställe (alltså även den nya observationen), skriv även ut antalet gånger de gjort observationen, och datum för den senaste av dessa observationer. Sortera medlemmarna efter när de gjorde sin senaste observation (den med mest aktuell observation skall komma först).
>
> Om något går fel, exempelvis att platsen eller arten inte finns, eller att vi redan har tre observationer av arten på platsen denna dag (se uppgift 4), så skall vi skriva ut ett lämpligt felmeddelande, och avbryta (vi behöver då inte skriva ut listan med observationer).

Databasen ligger i SQLite-filen `"birds.sqlite"`.

Du skall skriva ett program som går att köra, men du behöver inte skriva några import-satser. Om du skriver Pythonkod måste du använda `sqlite3`-biblioteket, och om du skriver Javakod måste du använda `JDBC`-paketet.

För att läsa in en sträng kan du använda följande anrop:

- I Python: `s = input("Ange s: ")`
- I Java: `var s = EDAF75.nextLine("Ange s: ");`

*4 p*

## Uppgift 6

(a) Förklara vad som är det grundläggande problemet när man vill ha en relationsdatabas för släktforskning. Vilken teknik kan vi använda istället, och hur hjälper den oss? *1.5 p*

(b) Förklara vad bokstäverna i akronymen ACID står för, och förklara kortfattat hur de hjälper oss när vi arbetar med en databas. *1.5 p*